PUMPKIN™
SPACE SYSTEMS

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

GUTS Software Description

HISTORY

Pumpkin's GUTS flight software is a heavily modified fork of the KubOS flight software. Originally, KubOS was chosen for a 12U-Optics Bus project, however, many improvements to the core of the flight software were made. In addition, many of the services/libraries originally provided from KubOS were overhauled or replaced to better fit the requirements for both Pumpkin's SUPERNOVA Architecture, as well as the 12U-Optics Bus mission requirements. There were sufficient changes to declare the Pumpkin's fork of KubOS as a different FSW, hence the new Pumpkin GUTS flight software name.

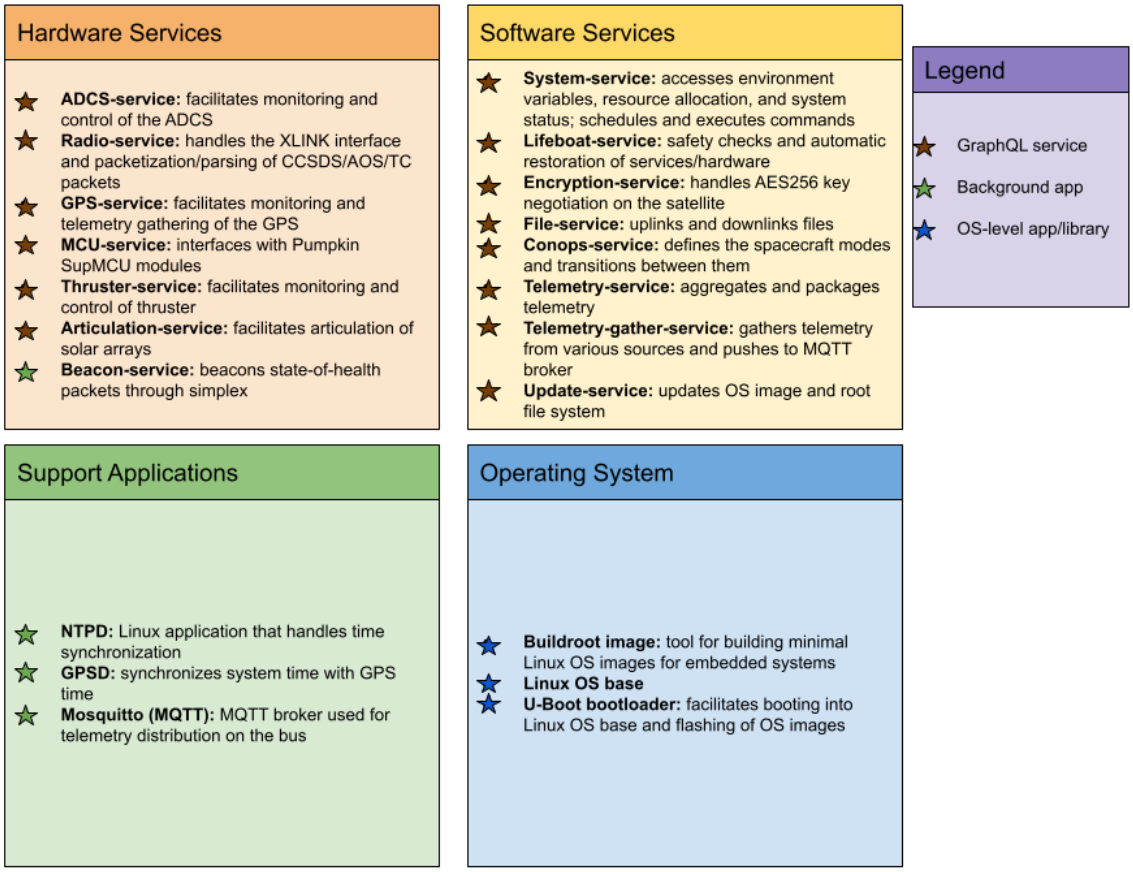


Figure 1 - GUTS FSW Services/Libraries/Software

BASIC WORKING PRINCIPLES

The GUTS flight software includes a `yocto` image, Linux OS base, and a U-Boot bootloader with flight software services running on top. The Linux operating system uses an EXT4 file system and supports a default `bash` CLI.

Any given GUTS flight software service represents an independent function of the satellite, such as an ADCS service to control the satellite's ADCS, and is written in Rust primarily with options for Python-based services. The GUTS services use a GraphQL interface to allow querying information from the satellite and sending commands, known as mutations. Each service binds to a local TCP/IP port on the Linux OS, and responds to HTTP POST requests. For requests from the Ground, the Radio Service facilitates receiving encrypted CCSDS packets containing a GraphQL Request over RF/Ground-Link to make on the behalf of the S/C operator. For commanding, Pumpkin uses Stored Command Sequences (.scs files) which is a set of GraphQL requests to make serially. The GUTS FSW Services are user-binaries, allowing updates through simply copying the new software onto the satellite to replace the existing version.

The GUTS flight software is a highly configurable platform, allowing the user to leverage only the features needed, and select capabilities required for a given customer's mission. A large subset of parameters are settable via text-based configuration files (TOML), allowing for small (<64k) on-orbit updates to how the bus exports telemetry, behavior of modes and a plethora of other configurable parameters. In addition, GUTS supports discovery of SUPERNOVA Bus modules (e.g. EPSM) to automatically reconfigure the available telemetry/commands based off of SUPERNOVA's hardware configuration.

Figure 2 - GraphQL Playground

COMMUNICATIONS

Application data is sent in CCSDS packets as specified in CCSDS 133.0-B-2. Uplinked files, commands and payload data are sent in TC Transfer Frame format (CCSDS 232.0-B-3) with TC Synchronization and Channel Coding (CCSDS 231.0-B-3). Downlinked files, command responses, and payload data are sent in AOS Transfer Frames (CCSDS 732.0-B-3) with TM Synchronization and Channel Coding (CCSDS 131.0-B-3).

Communication from the payload to ground is handled through data ports that allow the user to send arbitrary CCSDS packets. The payload can connect to a data port, typically port 8887/tcp, on the bus and send CCSDS packets to be transmitted to the ground. The radio-service marks the generated 2048-byte AOS frames as real-time or orbital data based on the associated CCSDS APID and configuration values.

The GUTS FSW supports ground-to-payload communication by forwarding assigned APIDs to a given IP address and port with frames sent via TCP/IP. The bus will connect to the payload, stream any CCSDS packets to be sent, and then disconnect. All the streamed CCSDS packets shall be in the original format they were sent up, the `radio-service` serves as a direct passthrough of CCSDS packets to/from the payload.

The GUTS flight software supports an IQ Wireless XLINK radio. The `radio-service` facilitates all communication to and from the IQ Wireless XLINK radio, and validates the radio is properly configured. Whenever the XLINK radio settings fall out of sync (e.g. on startup or restart of the radio), the `radio-service` resyncs the proper RF configuration and downlink rate parameters. The `radio-service` supports reconfiguration of the following on the IQ Wireless XLINK Radio:

- Data downlink rate (1Mbps -> 55 Mbps on X-band radio)
- TX and RX Frequencies
- TX Output power and attenuation
- TX Output modulation (BPSK, QPSK, 3-8PSK, 4-16PSK, 5-32PSK, 6-64PSK)
- TX Forward Error Correction encoding

Other configuration parameters such as RX Modulation can be requested when ordering IQ Wireless XLINK Radio.

PUMPKIN™
SPACE SYSTEMS

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

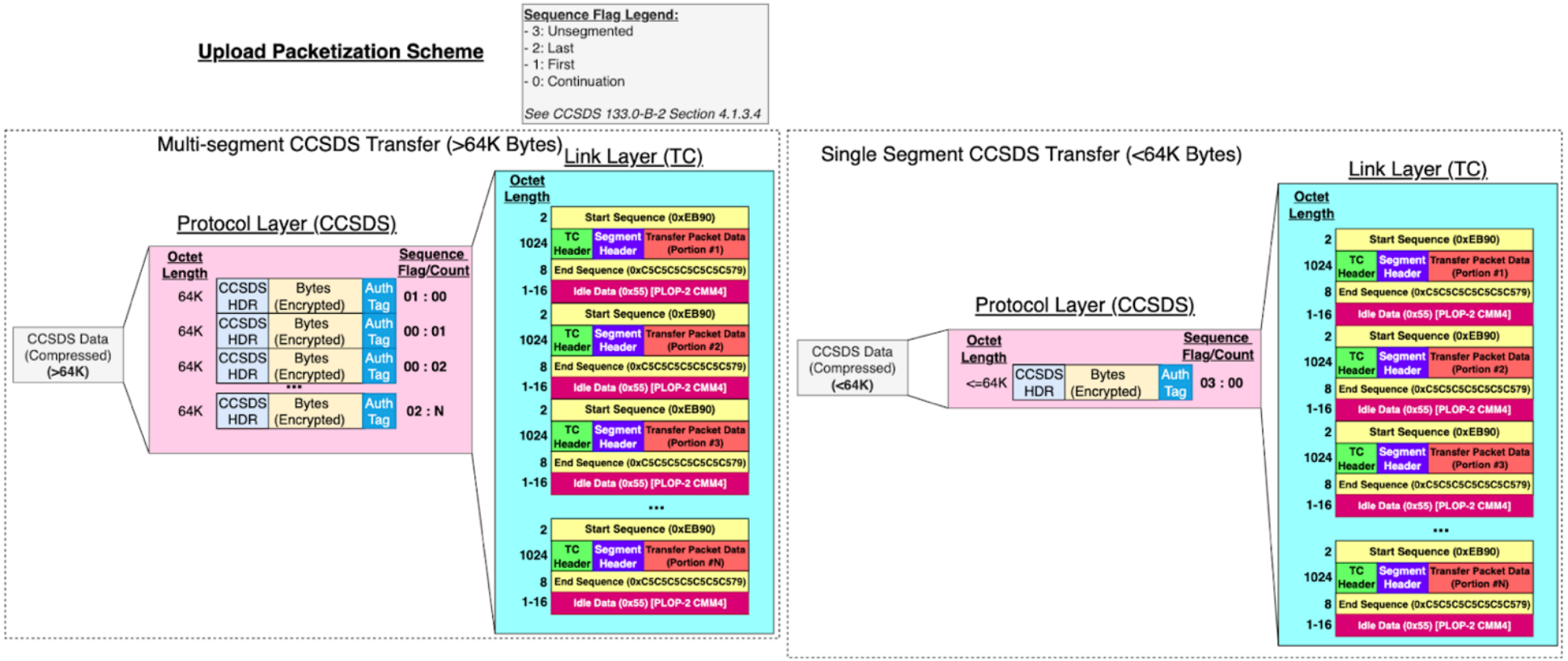


Figure 3 - Uplink Packetization for CCSDS Packets from Ground

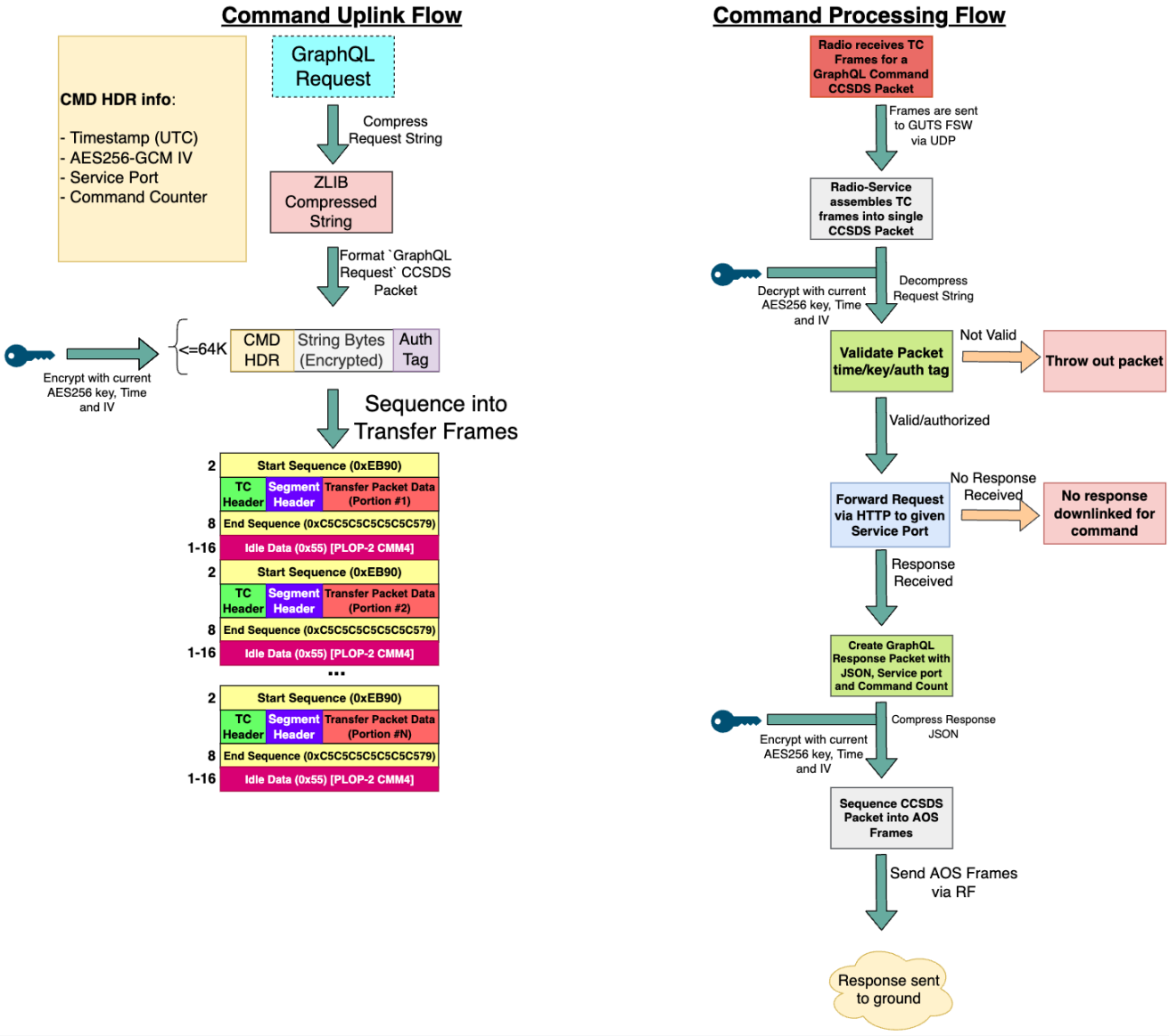
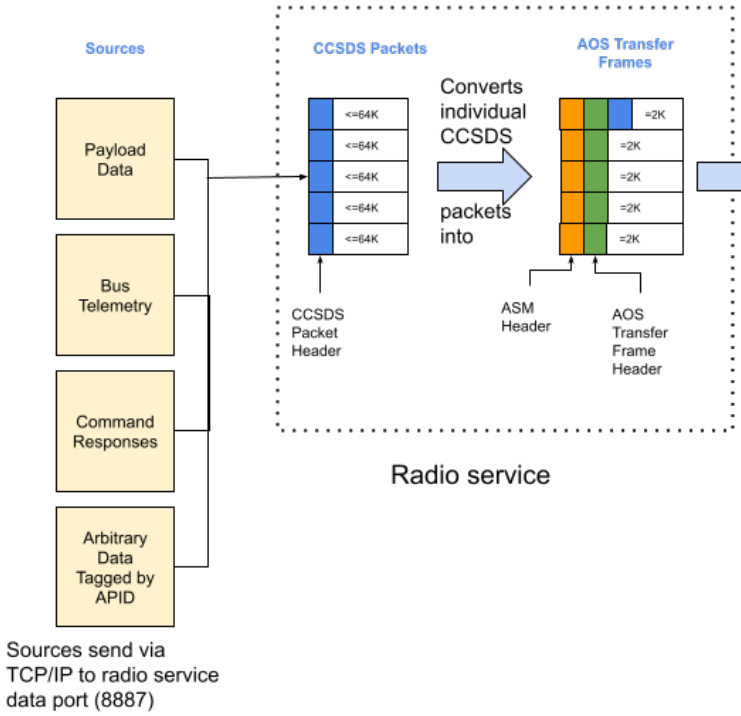
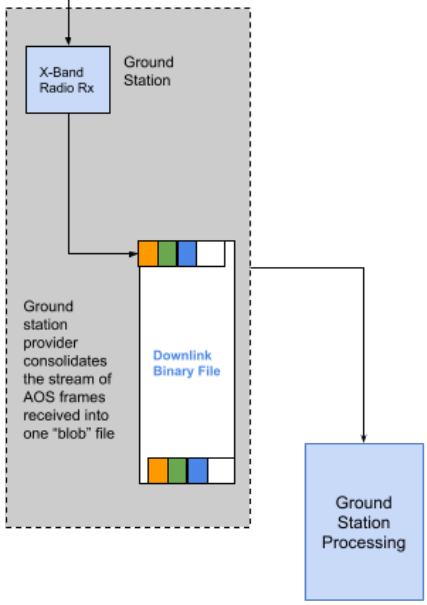


Figure 4 - GraphQL Command Uplink Flow & Processing

GUTS Downlink Data — Packetization Flow





AOS Transfer Frame Format

Note:

The Transfer Frame format used for data downlink. The format shall follow the spec as listed in **CCSDS 732.0-B-3 Section 4.1.2** and the **Transfer Frame Data Field** shall use **Multiplexing Protocol Data Unit** (M_PDU) for each underlying **Virtual Channel ID** (VCID).

Note: The framing shall follow the TM Coding spec as listed in CCSDS 131.0-B-3 using the ASM Bit Pattern '0x1ACFFC1D' as listed in section 9.3.5.

Byte	Description	Value	Data Type	Group
0 - 3	Start Sync Marker	0x1ACFFC1D	u32	Sync
4	7:6 - Transfer Frame Version	1	u8	AOS Primary Header
	5:0 - SCID (MSB)	--		
5	7:6 - SCID (LSB)	0	u8	
	5:0 - VCID	0 = Orbit Data, 1 = Real-time Data		
6-8	Virtual Channel Frame Count	--	u8	
9	7 - Replay Flag	0	u8	
	6 - VC Frame Count Usage	0		
	5:4 - Reserved	0		
	3:0 - VC Frame Count Cycle	0		
10 - 11	15:11 - Reserved (Spare)	0	u16	M_PDU Header
	10:0 - First Header Pointer	See CCSDS 732.0-B-3 Section 4.1.4.2.3		
M - 2052	File CCSDS Packets	See CCSDS 732.0-B-3 Section 4.1.4.2.4	u8	M_PDU Packet Zone

GUTS CCSDS Data File Format

APID: **XXX -- Depends on File/Data type**

Note: **File Data CCSDS Packet** -- Shown below is Unsegmented data. Chunk data can be spread across multiple CCSDS Packets

Byte	Description (Endianness)	Value	Data Type	Group
0	7:5 - CCSDS Version	0	u8	CCSDS Primary Header
	4 - Packet Type	0		
	3 - Secondary Header Flag	1		
	2:0 - APID MSB	XXX		
1	APID LSB	XXX	u8	
2	7:6 - Sequence Flags	3	u8	
	5:0 - Packet Sequence Count MSB	--		
3	Packet Sequence Count LSB	--	u8	
4 - 5	Packet Data Length (Value is N-1)	--	u16	
6-21	AES256-GCM IV	--	u8[]	CCSDS Secondary Header (Unencrypted)
22-29	UTC J2000 Timestamp [ms] (Big Endian [MSB])	--	u64	CCSDS Secondary Header (Encrypted)
30-33	File ID (Big Endian [MSB])	--	u32	
34-49	BLAKE2s-128 Unencrypted File Hash (Big Endian [MSB])	--	u8[]	
50-(N-16)	Complete Compressed File Data	--	u8[]	CCSDS User Data Field (Encrypted)
(N-15)-N	AES256-GCM Auth Tag	--	u8[]	CCSDS User Data Field (Unencrypted)

Figure 5 - CCSDS Downlink Packetization and File Formats

System Safety

The lifeboat-service can monitor any telemetry value on the bus that can be queried via GraphQL as well as monitor program output and return codes. The service allows the user to configure areas of interest to regularly check, such as ensuring ADCS sensors are not frozen and that communication through the radio is maintained, and an action to take if a check fails. Multiple stages of recovery can be defined, such as trying

If the lifeboat-service determines that a subsystem needs to be recovered, it will take the action outlined by the user for the affected area. Possible actions include program execution, execution of a stored command sequence, rebooting the operating system, or power cycling the bus. Information about recoveries, including the number of recoveries attempted, areas affected, and the exact action taken to recover the bus are logged. The lifeboat-commander, a helper application for the lifeboat-service, provides an interactive TUI to examine historical recovery events.

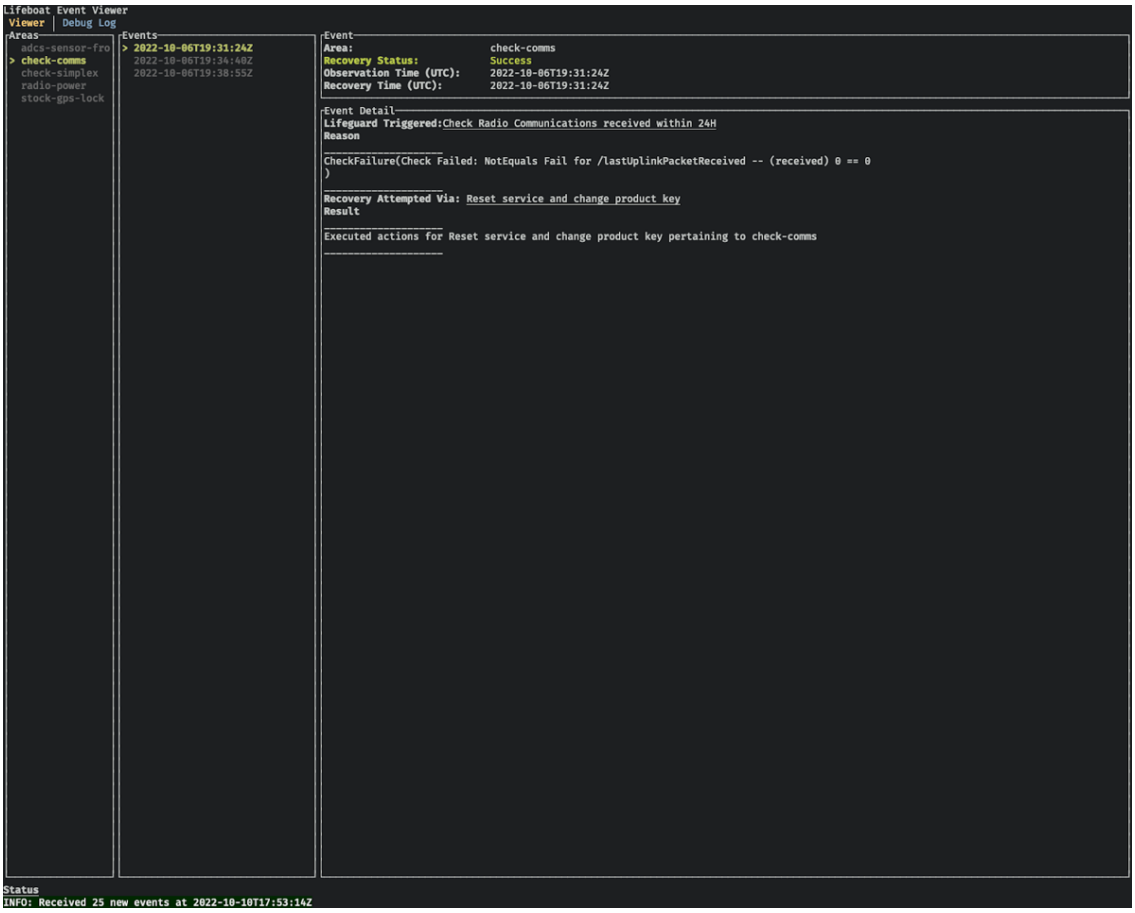


Figure 6 - Lifeboat Event Viewer

ENCRYPTION

The encryption-service handles AES256 key negotiation. The cipher used is AES256-GCM. The service can install new AES256 keys on the satellite, verify the origin of keys against the ground signing key installed on the spacecraft, query the active AES key bytes, and change the current encryption key to use. The bus encryption key is a RSA-2048 bit private key stored on the spacecraft. The ground signing key is to be used by S/C operators to sign a given AES256 key with a SHA-512 digest. The ground signing key and the public key of the bus are both needed to change the bus encryption key to validate the request as coming from an authenticated ground user. All communications to the SUPERNOVA Bus must be encrypted.

The GUTS FSW handles encryption for GraphQL requests over RF and file downlinks from the bus (e.g. telemetry files), using the current encryption key activated by the operator. All files, command responses and communications directly from the GUTS FSW are encrypted.

The data to/from the Payload is encrypted, and the GUTS FSW works as a pass-through for CCSDS packets based off APID. The Payload is to query the GUTS FSW Encryption Service for the current AES256 keys for uplink/downlink, then resolve encryption directly within the Payload.

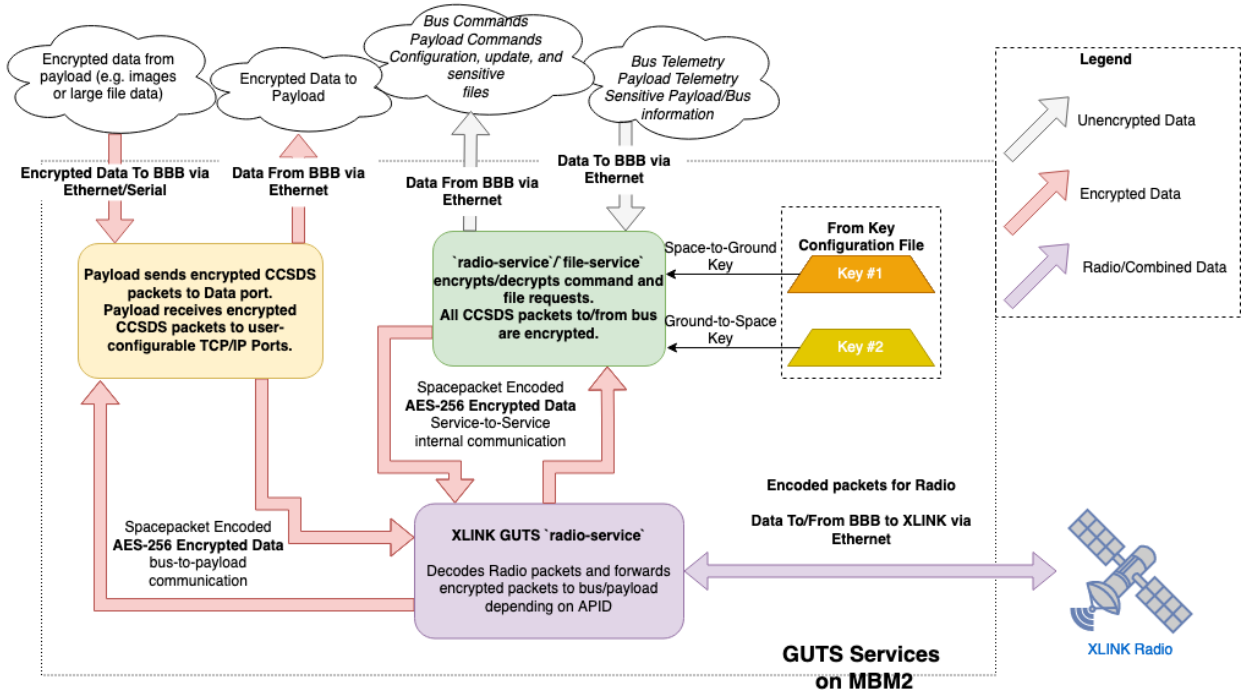


Figure 7 - Encryption Data Flow

TELEMETRY CAPABILITIES

The Pumpkin GUTS flight software features a rich telemetry set and configuration system allowing the user to choose the exact telemetry rate, items and format needed. All GUTS flight software services can export telemetry via a `telemetry-gather-service` running in the GUTS FSW. The gather service pushes user-specified GraphQL Queries to the `mosquitto` MQTT broker running on the bus, allowing for the Payload, Bus `telemetry-service` and ground-testing software (via script pushing telemetry data from MQTT to InfluxDB and displaying on Grafana).

The `bus-telemetry-service` allows the operator to export one or more telemetry streams for on-orbit retrieval. The `bus-telemetry-service` allows the user to set the collection rate (in seconds), how much telemetry to store in MB, and output format (either JSON or user-definable Binary format). This service also allows the operator to request the historical telemetry gathered on the bus, up to a number of days determined by how much historical telemetry to save in MB.

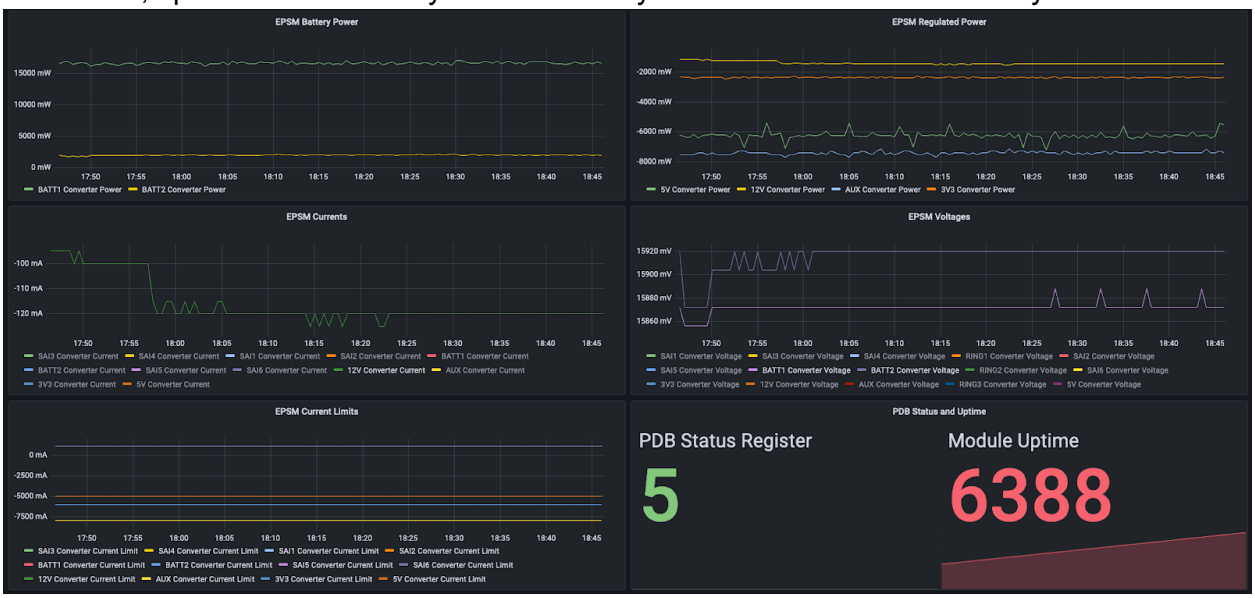


Figure 8 - Grafana Display from Environmental Testing

HARDWARE SERVICES AVAILABLE

- The **ADCS-service** facilitates monitoring and control of the ADCS. It runs two threads, one to maintain synchronization of time and GPS coordinates and another to carry out GraphQL requests.
- The **radio-service** handles the XLINK interface and packetization/parsing of CCSDS/AOS/TC packets. The *radio-service-daemon* executable runs on the ground station and handles multiplexing GraphQL and file requests through a simulated XLINK radio. Additionally, a lower-level *fake-xlink-radio* binary is available for the user to directly send TC frames or receive AOS frames via TCP/IP.
- The **GPS-service** facilitates monitoring and telemetry gathering of the NovAtel GPS. The GPS NMEA strings are mirrored directly to the gpsd binary running on the GUTS flight software.
- The **MCU-service** interfaces with all Pumpkin SupMCU modules. It allows sending of commands to specific modules, such as enabling power ports, and querying module telemetry. It also regularly kicks the watchdog timer on the EPSM.
- The **thruster-service** facilitates monitoring and control of the thruster. It can write parameters, read telemetry, toggle whether the thruster is enabled, and command the thruster to fire.
- The **beacon-service** beacons state-of-health packets through Simplex. All beacon rates can be specified via configuration from the user.
- The **articulation-service** facilitates articulation of solar arrays. Features include:
    - Query the current state of the DASA unit,
    - Home the solar array automatically.
    - Articulation modules include:
        - Hold a specific degree position
        - Track the MEEUS sun vector continuously based off of current S/C orientation
        - Follow a series of discrete steps
        - Turn off automatic articulation and allow manual commands.
    - *The flowchart below outlines the control loop for the articulation service.*

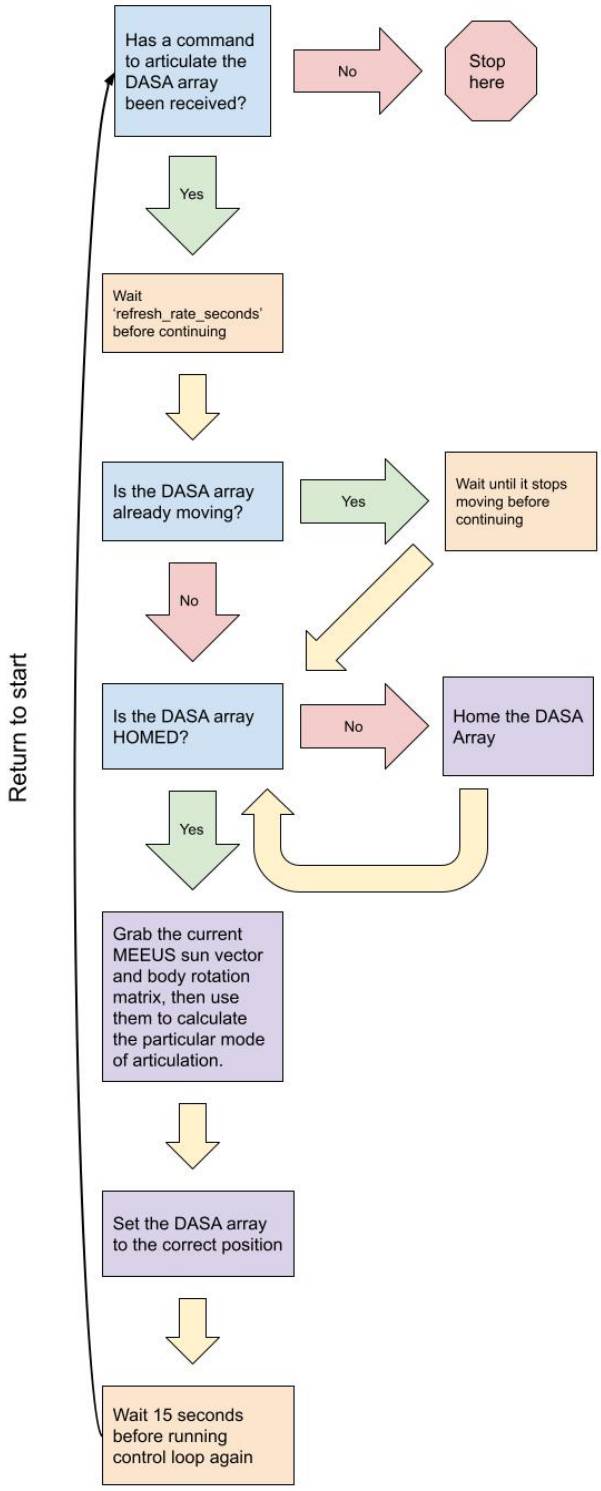


Figure 9 - DASA Articulation Logic

SOFTWARE SERVICES AVAILABLE

- The **system-service** accesses environment variables, resource allocation, and system status. It also executes arbitrary executables and scripts on the bus and schedules execution of commands and stored command sequence lists.
- The **lifeboat-service** performs safety checks and automatic restoration of services and hardware. For more information, refer to the section on system safety.
- The **encryption-service** handles AES256 key negotiation on the satellite. For more information, refer to the section on encryption.
- The **file-service** uplinks and downlinks arbitrary files. It features retransmission of files based on File ID, a GraphQL endpoint for starting uplink and downlink of files, compression and decompression of files via zLib, usage of the BLAKE2s hash for file verification, and usage of encryption parameters.
- The **CONOPS-service** defines the spacecraft modes and transitions between them. The currently supported modes are Deploy, Idle, Science, Downlink, and Safe.
- The **telemetry-service** aggregates and packages telemetry. Telemetry can come from the GUTS Telemetry Database or from a dummy source for testing purposes and be packaged as JSON data, binary data structured according to the service configuration, or raw output printed for testing purposes.
- The **telemetry-gather-service** gathers telemetry from various sources and pushes it to the MQTT broker.
- The **update-service** updates the OS image and root file system. It can also return information about the current version of the operating system and list available update files.

CONFIGURABILITY ON ORBIT/GROUND

Much of the operation of GUTS can be customized via configuration files. All configuration files are stored in one directory in the GUTS file system, the `/etc/guts.d`. All configuration files are stored in TOML format, and the GUTS FSW configuration can be edited via:

- Editing file via text-editor then uploading replacement configuration file over existing configuration on the bus
- Dedicated queries to GUTS flight software services (e.g. setting TX frequency of XLINK radio via `setTxFrequency` GraphQL Mutation).
- Dedicated configuration CLI arguments to service executables (e.g. setting XLINK product key via `radio-service configure set product-key`).

Examples of configuration include:

- The telemetry-service allows the user to choose which telemetry items are gathered and the rate at which they are gathered.
- The lifeboat-service includes functionality for users to define areas to monitor, the checks to perform for each area, and the recovery actions to take if needed.
- Startup commands to initialize the peripherals of the SUPERNOVA bus (e.g. configuring OEM719 for LEO orbit)
- The radio-service allows the operator to set XLINK radio transmit frequency, modulation, data rate, forward error correction encoding.

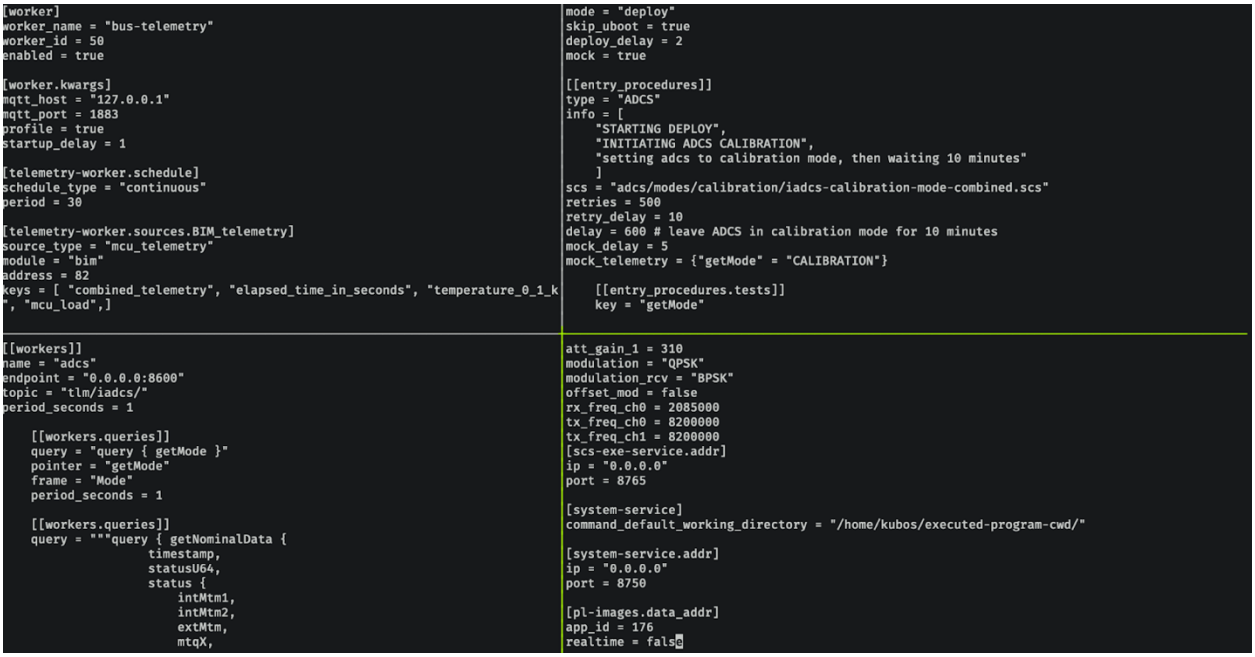


Figure 10 - GUTS Configuration File Examples